E-21.6VX
4

Design of Nonlinear Least-Squares Velocity Estimation Algorithms for Automotive Vehicles

David G. Taylor
Georgia Institute of Technology
School of Electrical and Computer Engineering
Atlanta, GA 30332-0250

1 Introduction

The purpose of this report is to document the initial stages of research in support of the development of a new velocity estimation algorithm for automotive vehicles. The velocity estimation problem may be described as follows. Suppose that a vehicle is represented as a rigid body moving in a plane with both translation and rotation. This motion is described by a linear velocity vector, consisting of a longitudinal component v_x and a lateral component v_y, and an angular velocity ω_z often referred to as yaw rate. The velocity estimation problem is the problem of estimating v_x, v_y and ω_z using signals from available sensors together with information describing the vehicle such as parameter values and mathematical equations. Some of the available sensors measure signals related to wheel motion, namely wheel rotational speeds and wheel steering angles. Other sensors that may be available measure signals that directly relate to motion of the vehicle body, namely the longitudinal and lateral components of acceleration and the yaw rate. The three signals that must be estimated are signals that are not directly measured by sensors, with the possible exception of yaw rate. The desire to estimate these signals is motivated by their usefulness in automatic control systems designed for safety enhancement.

Existing velocity estimation algorithms rely on several distinct principles. The most elementary velocity estimation algorithms use wheel speed sensors, steer angle sensors and a yaw rate sensor to estimate velocity vector components under an assumption of zero wheel slip. The resulting velocity estimation errors are proportional to the actual wheel slip, which means that these algorithms cannot perform accurately except when (i) at least one wheel operates with negligible slip and (ii) heuristic rules correctly identify which wheels operate with negligible slip. There exist many scenarios in which these requirements cannot generally be met, such as aggressive driving/braking and/or cornering, especially on slick road surfaces. Some apparently more sophisticated velocity estimation algorithms are actually layered algorithms in which the zero slip assumption is hidden in the first layer; examples of such layered estimation algorithms are [3], [2], [6] and [7]. The first three of these papers use polar coordinates for velocity vector representation, and they attempt to estimate the angle of the vector assuming that the length of the vector is known; the length of the velocity vector is not measured, but is supposed to be estimated using the zero slip assumption. The fourth of these papers uses cartesian coordinates for velocity vector representation, and it attempts to estimate the lateral component of the vector assuming that the longitudinal component of the vector is known; the longitudinal velocity component is not measured, but is supposed to be estimated using the zero slip assumption.

In contrast, the velocity estimation algorithm proposed in this report does not rely on a zero slip assumption in any way whatsoever. Instead, it relies on a model of the vehicle dynamics that, if properly calibrated, can yield accurate predictions of vehicle motion even when wheel slip is large. The proposed algorithm makes use of wheel speed sensors, steer angle sensors and some combination of body-mounted acceleration and yaw rate sensors. The condition that determines the suitability of a particular set of body-mounted sensors is driving along a straight path at constant speed (the worst-case scenario for preservation of observability). For the implementation reported herein, the minimum acceptable body-mounted sensor set is a pair of accelerometers for measuring longitudinal acceleration a_x and lateral acceleration a_y; the simulation example shown in a later section includes these two measurements as well as measurements of yaw rate ω_z. Although the proposed algorithm is more computationally complex than algorithms based on

zero slip assumptions, its real-time implementation should not be a problem considering the ever-increasing computational resources available in today's vehicles. Related dynamics-based velocity estimation algorithms that employ extended Kalman filters have been previously reported in the literature. For example, [4] and [5] use nonlinear vehicle dynamics but, to avoid the need for a tire model, these works require wheel torque sensors. In [1], the need for wheel torque sensors is removed by relying on a linearized tire model in which effective cornering stiffnesses are adapted on-line. In contrast to these existing dynamics-based methods, the proposed estimation algorithm makes use of a nonlinear tire model and is developed from the perspective of nonlinear least-squares optimization.

Since the proposed velocity estimation algorithm is based on a nonlinear model of vehicle dynamics, including a nonlinear tire model, it is logical to consider which of the parameter values found in these models are subject to change during operation of the vehicle. For the implementation reported herein, the parameter values appearing in the modeling equations of the estimator are: g, the acceleration of gravity; l_F, the distance from the CG to the front axle; l_R, the distance from the CG to the rear axle; h, the distance from the CG to the road surface; r, the wheel radius; m, the sprung mass; I_z, the sprung mass moment of inertia; T_s, the sampling period; (c_1, c_2, c_3), parameters of the nonlinear tire model; and μ, the road surface friction coefficient. Of these, the present implementation considers only μ to be unknown. Hence, a logical extension of this initial work would be to check the consequences of parameter mismatch for the parameters that have been assumed to be known, making modifications to the algorithm as necessary. Of the above cited works, only [4] and [5] account for unknown μ within the context of the velocity estimation problem, and those works have the disadvantage of wheel torque measurement in comparison to the proposed velocity estimation algorithm of this report.

2 Proposed Velocity Estimation Algorithm

The proposed velocity estimation algorithm for automotive vehicles is actually a nonlinear least-squares state estimation algorithm based on a discrete-time representation of the underlying system's continuous-time dynamics, expressed in the generic form

$$x_{k+1} = f(x_k, u_k) \tag{1}$$

$$y_k = h(x_k, u_k) \tag{2}$$

where u_k denotes the m-dimensional input vector at the kth sampling instant, x_k denotes the n-dimensional state vector at the kth sampling instant, and y_k denotes the p-dimensional output vector at the kth sampling instant. The functions f and h are generally nonlinear.

The state estimation problem may be stated as follows: determine an estimate $\hat{x}_k$ of the unmeasured x_k using the l-length measurement sets $\{u_k, u_{k-1}, \ldots, u_{k-l+1}\}$ and $\{y_k, y_{k-1}, \ldots, y_{k-l+1}\}$. To reduce the impact of sensor noise, the entire measurement sets should be used in the determination of $\hat{x}_k$. This suggests a two-step procedure, wherein first $\hat{x}_{k-l+1}$ is determined from the measurement sets and second $\hat{x}_k$ is determined through forward propagation.

According to (1)–(2), the value of x_{k-l+1} is algebraically related to the measurement sets through a nesting of functions. To present the algebraic relationship most clearly, it is useful to introduce the notations $f^u(x) := f(x, u)$ and $h^u(x) := h(x, u)$. With these notations, it is straightforward to show that

$$\underbrace{\begin{bmatrix} y_{k-l+1} \\ y_{k-l+2} \\ \vdots \\ y_k \end{bmatrix}}_{Y_k} = \underbrace{\begin{bmatrix} h^{u_{k-l+1}}(x_{k-l+1}) \\ h^{u_{k-l+2}} \circ f^{u_{k-l+1}}(x_{k-l+1}) \\ \vdots \\ h^{u_k} \circ f^{u_{k-1}} \circ \cdots \circ f^{u_{k-l+1}}(x_{k-l+1}) \end{bmatrix}}_{H(x_{k-l+1}, U_k)} \tag{3}$$

where U_k and Y_k denote the measurement sets in vector form, H denotes a mapping from state and input values to output values, and $\circ$ denotes composition of functions. In the absence of modeling error and sensor noise, the algebraic relationship $Y_k = H(x_{k-l+1}, U_k)$ must hold true; however, in reality there generally does not exist any value of x_{k-l+1} that perfectly satisfies this algebraic relationship. The measurement vectors

U_k and Y_k are known, so (3) represents l constraints on n unknown state variables. Assuming $l \geq n$, it is reasonable to approximate x_{k-l+1} by the value of $\hat{x}_{k-l+1}$ that minimizes any residual error in satisfying the idealized algebraic relationship. For this reason, an appropriate state estimate is one that minimizes the sum of the squares of the residual errors, i.e.

$$\hat{x}_{k-l+1} = \arg\min \|Y_k - H(x, U_k)\|^2 \tag{4}$$

This least-squares state estimate is l samples delayed in time, so it is propagated forward in time using the nominal system dynamics to obtain

$$\hat{x}_k = f^{u_{k-1}} \circ \cdots \circ f^{u_{k-l+1}}(\hat{x}_{k-l+1}) \tag{5}$$

In summary, $\hat{x}_k$ is the desired estimate of x_k obtained by way of (4) and (5).

The key step in the computational process is a nonlinear least-squares problem requiring an iterative search for the minimizer of $\|E_k(x)\|^2$, where $E_k(x) = Y_k - H(x, U_k)$ denotes the residual error at sample index k and H denotes the nonlinear map relating unmeasured state variables to sensor measurements. For the implementation described in this report, the iterative search is conducted according to the damped Gauss-Newton method developed as follows. Suppose that the current iterate in the search for the minimizer of $\|E_k(x)\|^2$ is denoted by $x^{(j)}$. Using Taylor series approximation, a corresponding local model for the iteration process would be

$$\hat{E}_k(x) = E_k(x^{(j)}) + J_k(x^{(j)})(x - x^{(j)}) \tag{6}$$

where $J_k(x)$ denotes the Jacobian matrix of $E_k(x)$ with respect to x. If $x^{(j)}$ is close to the minimizer, then a logical way of generating the next iterate $x^{(j+1)}$ would be to insist that $\hat{E}_k(x^{(j+1)})$ have minimum norm. The minimum norm solution is well defined if the Jacobian matrix has full column rank, and in that case it may be computed using

$$x^{(j+1)} = x^{(j)} - \left(J_k(x^{(j)})^T J_k(x^{(j)})\right)^{-1} J_k(x^{(j)})^T E_k(x^{(j)}) \tag{7}$$

However, since $x^{(j)}$ may not be very close to the minimizer, the application of Taylor series approximation may not lead to a sufficiently accurate local model, and therefore some modification is needed. Fortunately, it can be shown that whenever the step increment defined by (7) is well defined it is guaranteed to be in a descent direction, so a sufficiently small step in the direction of the step increment defined by (7) is guaranteed to reduce the residual error. Consequently, the implementation of the iterative search process employed in this report is based on the recursion

$$x^{(j+1)} = x^{(j)} - \gamma^{(j)} \left(J_k(x^{(j)})^T J_k(x^{(j)})\right)^{-1} J_k(x^{(j)})^T E_k(x^{(j)}) \tag{8}$$

where $0 < \gamma^{(j)} \leq 1$ is a scalar parameter that may be used to limit the length of each step as needed. The iteration (8) is known in the literature as the damped Gauss-Newton method for nonlinear least-squares problems: it can be made globally convergent by appropriate choice of damping parameter $\gamma^{(j)}$, and it provides a quadratic rate of convergence near solutions with $\gamma^{(j)} = 1$.

3 Modeling Required for Algorithm Implementation

The proposed velocity estimation algorithm is based on the general methodology outlined in the previous section. It relies on a discretized vehicle dynamics model for predicting the effect of measured wheel speeds and measured steer angles on vehicle motion (according to vector u), with corrective action provided by body-mounted sensors (according to vector y). As implemented in this report, the discretization is based on the forward Euler method of numerical integration. In particular, suppose that the continuous-time system dynamics are modeled in the generic form

$$\dot{x}(t) = f_c(x(t), u(t)) \tag{9}$$

$$y(t) = h_c(x(t), u(t)) \tag{10}$$

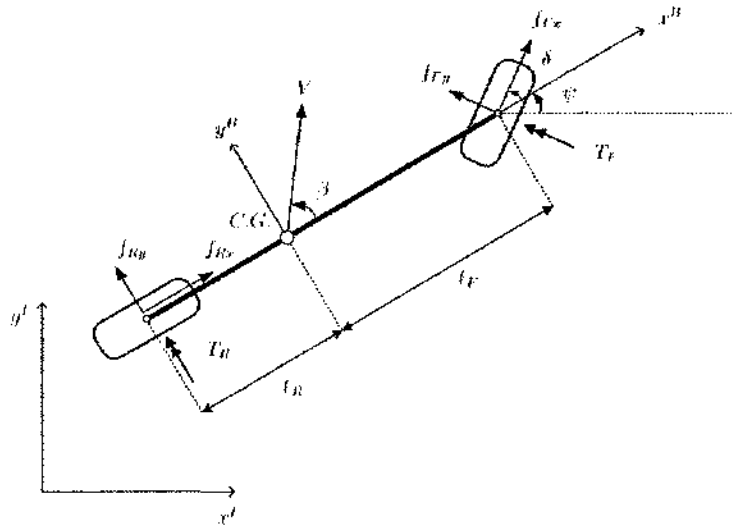


Figure 1: Sign conventions and parameter values for the vehicle dynamics model (shown specialized to front-wheel steer case) that is used as the basis of the proposed velocity estimation algorithm.

where t is time, and $u(t)$, $x(t)$ and $y(t)$ represent the time trajectories of the input vector, state vector and output vector, respectively. By definition, the first time derivative of $x(t)$ is given by

$$\dot{x}(t) = \lim_{T \to 0} \frac{x(t+T) - x(t)}{T} \tag{11}$$

As a consequence, for sufficiently small T it follows that

$$x(t+T) \approx x(t) + T\dot{x}(t) = x(t) + T f_c(x(t), u(t)) \tag{12}$$

which leads to the corresponding (approximate) discrete-time model

$$x_{k+1} = f(x_k, u_k) \tag{13}$$
$$y_k = h(x_k, u_k) \tag{14}$$

where

$$f(x_k, u_k) = x_k + T_s f_c(x_k, u_k) \tag{15}$$
$$h(x_k, u_k) = h_c(x_k, u_k) \tag{16}$$

and $u_k = u(kT_s)$, $x_k = x(kT_s)$ and $y_k = y(kT_s)$ with sampling period T_s. The remainder of this section discusses specific modeling details used for algorithm implementation, based on the diagram of Fig. 1.

3.1 Motion Equations

Application of Newton's Laws with respect to the inertial frame yields equations of motion in the form

$$m\ddot{x} = f_{Fx}\cos(\delta_F + \psi) - f_{Fy}\sin(\delta_F + \psi) + f_{Rx}\cos(\delta_R + \psi) - f_{Ry}\sin(\delta_R + \psi) \tag{17}$$
$$m\ddot{y} = f_{Fx}\sin(\delta_F + \psi) + f_{Fy}\cos(\delta_F + \psi) + f_{Rx}\sin(\delta_R + \psi) + f_{Ry}\cos(\delta_R + \psi) \tag{18}$$
$$I_z\ddot{\psi} = l_F(f_{Fx}\sin\delta_F + f_{Fy}\cos\delta_F) - l_R(f_{Rx}\sin\delta_R + f_{Ry}\cos\delta_R) \tag{19}$$

where, for sake of generality, steer angles are permitted on both wheels. The velocity estimation problem requires estimation of the velocity vector components v_x and v_y as represented in the body frame, as well

as the angular velocity or yaw rate ω_z. These variables are related to motion variables as represented in the inertial frame through the rotation transformation

$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} \tag{20}$$

Application of this transformation to the inertial-frame model yields the body-frame model

$$\dot{v}_x = m^{-1}(f_{Fx}\cos\delta_F - f_{Fy}\sin\delta_F + f_{Rx}\cos\delta_R - f_{Ry}\sin\delta_R) + \omega_z v_y \tag{21}$$

$$\dot{v}_y = m^{-1}(f_{Fx}\sin\delta_F + f_{Fy}\cos\delta_F + f_{Rx}\sin\delta_R + f_{Ry}\cos\delta_R) - \omega_z v_x \tag{22}$$

$$\dot{\omega}_z = I_z^{-1}\{l_F(f_{Fx}\sin\delta_F + f_{Fy}\cos\delta_F) - l_R(f_{Rx}\sin\delta_R + f_{Ry}\cos\delta_R)\} \tag{23}$$

This body-frame model is not complete, however, until augmented with a tire/road model that describes how input and state variables ultimately determine wheel forces.

3.2 Force Model

The force vector produced by each tire is modeled as the product of the associated friction coefficient vector and normal force. Expressed in terms of components in the wheel frames, these force vectors take the form

$$f_{Fx} = \mu_{Fx} f_{Fz} \tag{24}$$

$$f_{Fy} = \mu_{Fy} f_{Fz} \tag{25}$$

$$f_{Rx} = \mu_{Rx} f_{Rz} \tag{26}$$

$$f_{Ry} = \mu_{Ry} f_{Rz} \tag{27}$$

In the absence of suspension system dynamics, the normal forces are not independent of the driving/braking forces since, in order to keep both wheels on the road surface, a moment balance must be satisfied at all times in the pitch plane. The moment balance is conveniently characterized in terms of inertial force

$$ma_x = f_{Fx}\cos\delta_F - f_{Fy}\sin\delta_F + f_{Rx}\cos\delta_R - f_{Ry}\sin\delta_R \tag{28}$$

Both wheels will remain grounded if the normal forces satisfy the moment balance constraints

$$f_{Fz}(l_F + l_R) - mgl_R = -ma_x h \tag{29}$$

$$f_{Rz}(l_F + l_R) - mgl_F = ma_x h \tag{30}$$

Hence, there are a total of six constraint equations governing six components of force associated with two wheels. The resulting constraint system is linear, and has been solved in closed form to obtain an explicit relationship between the normal forces and the friction coefficients (independent of driving/braking forces): this explicit relationship is derived in the appendix and is used in the Matlab code implementation. This novel modeling feature, apparently not found in the standard texts or in published papers, represents an explicit way of embedding load transfer effects into the vehicle dynamics model. This is advantageous for the proposed velocity estimation algorithm, which requires multiple on-line evaluations of the vehicle dynamics model each time a damped Gauss-Newton iteration is performed, since the explicit normal force solution employed avoids the need for on-line matrix inversion.

3.3 Friction Model

The friction coefficient vectors, needed to compute the driving/braking forces and the normal forces, are colinear with the corresponding relative velocity vectors. The concept of relative velocity arises since each wheel defines a velocity vector according to its rotational speed and its steering angle, but each wheel also travels with the vehicle body according to the velocity vector of the body at the point of attachment: the two velocity vectors associated with each wheel are generally distinct, and their difference is referred to as the relative velocity vector. When normalized by reference velocities, these relative velocity vectors become

dimensionless vectors and are referred to as slip vectors. The friction coefficient vectors depend on these slip vectors in a nonlinear fashion and are defined functionally by

$$\mu_{Fx} = \mu\left(c_1\left(1-e^{-c_2\|s_F\|}\right)-c_3\|s_F\|\right)\frac{s_{Fx}}{\|s_F\|} \quad (31)$$

$$\mu_{Fy} = \mu\left(c_1\left(1-e^{-c_2\|s_F\|}\right)-c_3\|s_F\|\right)\frac{s_{Fy}}{\|s_F\|} \quad (32)$$

$$\mu_{Rx} = \mu\left(c_1\left(1-e^{-c_2\|s_R\|}\right)-c_3\|s_R\|\right)\frac{s_{Rx}}{\|s_R\|} \quad (33)$$

$$\mu_{Ry} = \mu\left(c_1\left(1-e^{-c_2\|s_R\|}\right)-c_3\|s_R\|\right)\frac{s_{Ry}}{\|s_R\|} \quad (34)$$

where c_1, c_2 and c_3 are constant coefficients that determine the shape of the friction nonlinearities to match nominal tire/road characteristics, μ is a scale factor used to exhibit the effect of road surface changes, and the slip vectors

$$s_{Fx} = \frac{r\omega_F - v_{Fx}}{v_{Fx}} \quad (35)$$

$$s_{Fy} = \frac{-v_{Fy}}{v_{Fx}} \quad (36)$$

$$s_{Rx} = \frac{r\omega_R - v_{Rx}}{v_{Rx}} \quad (37)$$

$$s_{Ry} = \frac{-v_{Ry}}{v_{Rx}} \quad (38)$$

are computed from wheel rotational velocities, $r\omega_F$ and $r\omega_R$, and the velocity vectors of the vehicle body at the wheel locations as expressed by

$$v_{Fx} = v_x\cos\delta_F + (v_y + l_F\omega_z)\sin\delta_F \quad (39)$$

$$v_{Fy} = -v_x\sin\delta_F + (v_y + l_F\omega_z)\cos\delta_F \quad (40)$$

$$v_{Rx} = v_x\cos\delta_R + (v_y - l_R\omega_z)\sin\delta_R \quad (41)$$

$$v_{Ry} = -v_x\sin\delta_R + (v_y - l_R\omega_z)\cos\delta_R \quad (42)$$

With this nonlinear friction model, the friction coefficient vectors will be confined to so-called friction circles. The maximum lengths of the friction coefficient vectors are

$$\mu^* = \mu\left(c_1 - \frac{c_3}{c_2}\left(1-\ln\left(\frac{c_3}{c_1c_2}\right)\right)\right) \quad (43)$$

and they occur when the slip vectors have lengths equal to

$$s^* = -\frac{1}{c_2}\ln\left(\frac{c_3}{c_1c_2}\right) \quad (44)$$

This nonlinear friction model is incorporated without approximation into the proposed velocity estimation algorithm as shown. To account for situations in which the vehicle body is motionless, e.g. prior to launch, nonzero normalizing velocities would need to be used to determine slip.

3.4 Sensor Models

In the initial implementation and simulation, all sensors including accelerometers are modeled as ideal sensors. Accelerometer output signals are computed from state and input signals according to

$$a_x = m^{-1}(f_{Fx}\cos\delta_F - f_{Fy}\sin\delta_F + f_{Rx}\cos\delta_R - f_{Ry}\sin\delta_R) \quad (45)$$

$$a_y = m^{-1}(f_{Fx}\sin\delta_F + f_{Fy}\cos\delta_F + f_{Rx}\sin\delta_R + f_{Ry}\cos\delta_R) \quad (46)$$

The effects of sensor noise and bias should be studied in future work.

4 Matlab Code Implementation

This section contains a listing of the code used to implement the proposed velocity estimation algorithm. By comparing the code to the mathematical descriptions given earlier, it should be possible to grasp the essential ideas with maximum clarity.

4.1 Notations

g	g	m/s^2	acceleration of gravity
lf	l_F	m	distance from CG to front axle
lr	l_R	m	distance from CG to rear axle
h	h	m	distance from CG to road surface
r	r	m	wheel radius
m	m	kg	sprung mass
Iz	I_z	kg m^2	sprung mass moment of inertia
Ts	T_s	s	sampling period
c1	c_1	—	parameter of nonlinear tire model
c2	c_2	—	parameter of nonlinear tire model
c3	c_3	—	parameter of nonlinear tire model
mu	μ	—	road surface friction coefficient
df	δ_F	rad	front steer angle
dr	δ_R	rad	rear steer angle
wf	ω_F	rad/s	front wheel speed
wr	ω_R	rad/s	rear wheel speed
vx	v_x	m/s	component of CG velocity vector along body longitudinal axis
vy	v_y	m/s	component of CG velocity vector along body lateral axis
ax	a_x	m/s^2	component of CG acceleration vector along body longitudinal axis
ay	a_y	m/s^2	component of CG acceleration vector along body lateral axis
wz	ω_z	rad/s	angular velocity (yaw rate)
vfx	v_{Fx}	m/s	component of velocity vector along front wheel longitudinal axis
vfy	v_{Fy}	m/s	component of velocity vector along front wheel lateral axis
vrx	v_{Rx}	m/s	component of velocity vector along rear wheel longitudinal axis
vry	v_{Ry}	m/s	component of velocity vector along rear wheel lateral axis
sfx	s_{Fx}	—	component of slip vector along front wheel longitudinal axis
sfy	s_{Fy}	—	component of slip vector along front wheel lateral axis
srx	s_{Rx}	—	component of slip vector along rear wheel longitudinal axis
sry	s_{Ry}	—	component of slip vector along rear wheel lateral axis
mfx	μ_{Fx}	—	component of friction coefficient vector along front wheel longitudinal axis
mfy	μ_{Fy}	—	component of friction coefficient vector along front wheel lateral axis
mrx	μ_{Rx}	—	component of friction coefficient vector along rear wheel longitudinal axis
mry	μ_{Ry}	—	component of friction coefficient vector along rear wheel lateral axis
ffx	f_{Fx}	N	component of friction force vector along front wheel longitudinal axis
ffy	f_{Fy}	N	component of friction force vector along front wheel lateral axis
frx	f_{Rx}	N	component of friction force vector along rear wheel longitudinal axis
fry	f_{Ry}	N	component of friction force vector along rear wheel lateral axis

ffz	f_{Fz}	N	front wheel normal force
frz	f_{Rz}	N	rear wheel normal force

4.2 Script File

The following script file performs several functions in sequence. The user selects one of two types of nonlinear least-squares estimators. The basic estimator assumes that μ is constant and known so it only has to estimate the velocity components. The extended estimator treats μ as an unknown state that is to be estimated along with the velocity components. The reason that two versions are implemented is that there is a trade-off involved: the basic estimator is computable at steady-state straight path driving conditions, but it relies on knowledge of road surface conditions; the extended estimator does not rely on knowledge of road surface conditions, but it is not computable at steady-state straight path driving conditions. Parallel implementation of the two versions would provide the desired combination of features. After user selections have been made, the vehicle dynamics model is solved to obtain the vehicle response to specified inputs, and this vehicle simulation is followed by execution of the nonlinear least-squares estimation algorithm and the zero slip assumption estimation algorithm. Finally, the results are presented in the form of graphical plots.

```
clear all, close all

global g lf lr h r m Iz Ts c1 c2 c3

g = 9.81;
lf = 1.355;
lr = 1.527;
h = 0.546;
r = 0.329;
m = 1666;
Iz = 3447;
Ts = 12.5e-3;

c1 = 1.2801;
c2 = 23.99;
c3 = 0.52;

X = [ ]; Y = [ ]; U = [ ]; T = [ ]; v = 10;

% Basic Estimator => mu is constant and known
% Extended Estimator => mu varies and is unknown
est = menu('Estimator Type','Basic','Extended');

% Open-Loop Transient => sinusoidal steering with acceleration
% Steady-State Straight => used to test for loss of observability
path = menu('Path Type','Open-Loop Transient','Steady-State Straight');

% simulate vehicle dynamics
if est == 1
    x = [ v ; 0 ; 0 ];
elseif est == 2
    x = [ v ; 0 ; 0 ; 1 ];
end
for t = 0 : Ts : 4
    if est == 2 & t == 2
        x(4) = 0.5;
```

```
    end
    if path == 1
        u = [ (30*pi/180)*sin(2*pi*t/2) ; 0 ; (v/r)*1.3 ; (v/r)*1.3 ];
    elseif path == 2
        u = [ 0 ; 0 ; (v/r)*1 ; (v/r)*1 ];
    end
    [xdot,y] = dynamics(x,u);
    X = [ X x ]; Y = [ Y y ]; U = [ U u ]; T = [ T t ];
    x = x+Ts*xdot;
end

% estimate vehicle velocities using NLS estimator
Xhat = [ ]; xhat = X(:,1); N = 8;
for k = N : length(T)
    uu = U(:,k-N+1:k);
    yy = Y(:,k-N+1:k);
    xhat = NLS_est(xhat,uu,yy);
    Xhat = [ Xhat xhat ];
end

% estimate vehicle velocities using ZSA estimator
VXhat = [ ]; VYhat = [ ];
for k = 1 : length(T)
    u = U(:,k);
    wz = X(3,k);
    [vxhat,vyhat] = ZSA_est(u,wz);
    VXhat = [VXhat vxhat]; VYhat = [VYhat vyhat];
end

% plot vehicle state trajectory
figure, plot(T,X)
xlabel('time'), ylabel('vehicle state trajectory')
if est == 1
    legend('v_x','v_y','\omega_z')
elseif est == 2
    legend('v_x','v_y','\omega_z','\mu')
end

% plot NLS estimator output
figure, plot(T(:,N:length(T)),Xhat)
xlabel('time'), ylabel('NLS estimator output')
if est == 1
    legend('v_x^{NLS}','v_y^{NLS}','\omega_z^{NLS}')
elseif est == 2
    legend('v_x^{NLS}','v_y^{NLS}','\omega_z^{NLS}','\mu^{NLS}')
end

% plot NLS estimation errors
Ehat = Xhat-X(:,N:length(T));
figure, plot(T(:,N:length(T)),Ehat)
xlabel('time'), ylabel('NLS estimation error')
if est == 1
    legend('v_x^{NLS}','v_y^{NLS}','\omega_z^{NLS}')
 elseif est == 2
```

```
    legend('v_x^{NLS}','v_y^{NLS}','\omega_z^{NLS}','\mu^{NLS}')
end

% plot ZSA estimator output
figure, plot(T,VXhat,T,VYhat)
xlabel('time'), ylabel('ZSA estimator output')
legend('v_x^{ZSA-F}','v_x^{ZSA-R}','v_y^{ZSA-F}','v_y^{ZSA-R}')

% plot ZSA estimation errors
EXhat = VXhat-[X(1,:);X(1,:)];
EYhat = VYhat-[X(2,:);X(2,:)];
figure, plot(T,EXhat,T,EYhat)
xlabel('time'), ylabel('ZSA estimation error')
legend('v_x^{ZSA-F}','v_x^{ZSA-R}','v_y^{ZSA-F}','v_y^{ZSA-R}')
```

4.3 Function Files for the Estimation Algorithms

4.3.1 Nonlinear Least-Squares Estimator

This estimation algorithm performs a fixed number of damped Gauss-Newton iterations in an effort to minimize the state estimation error. The required input data are an initial estimate of the state vector `xhat`, a matrix of input sensor values `uu`, and a matrix of output sensor values `yy`; the matrices `uu` and `yy` must have the same number of columns. The user specifies the desired number of iterations via `iterations` and the desired damping parameter via `stepsize`. The residual error vector is obtained by calling `Efun`, and the corresponding numerically approximated Jacobian matrix is obtained by calling `Jfun`. Each of the damped Gauss-Newton iterations requires solution of a linear least-squares problem, as implemented using the \ operator. Once all of the damped Gauss-Newton iterations have been performed, a selection is made between the final state estimate `xhat` or the initial state estimate `temp`, on the basis of which provides the smaller estimation error. The need for this selection is a consequence of allowing the user to pre-specify a fixed (perhaps insufficiently small) value for `stepsize`. Finally, the selected state estimate is updated to the present time by calling `dynamics` and iterating the state model an appropriate number of times.

Main Function

```
function xhat = NLS_est(xhat,uu,yy)

global Ts

iterations = 20;
stepsize = 1;
temp = xhat;

for j = 1 : iterations
    E = Efun(xhat,uu,yy);
    J = Jfun(xhat,uu,yy);
    xhat = xhat-stepsize*(J\E);
end

if norm(Efun(xhat,uu,yy)) > norm(Efun(temp,uu,yy))
    xhat = temp;
end

for k = 1 : length(uu)-1
    [xhatdot,yhat] = dynamics(xhat,uu(:,k));
    xhat = xhat+Ts*xhatdot;
```

```
end
```

Function to Evaluate Error Vector

```
function E = Efun(xhat,uu,yy)

global Ts

E = [ ];
for j = 1 : size(uu,2)
    u = uu(:,j);
    y = yy(:,j);
    [xhatdot,yhat] = dynamics(xhat,u);
    xhat = xhat+Ts*xhatdot;
    E = [ E ; y-yhat ];
end
```

Function to Evaluate Jacobian Matrix

```
function J = Jfun(xhat,uu,yy)

dx = 1e-12;
I = eye(length(xhat));
J = [ ];
for j = 1 : length(xhat)
    e = I(:,j);
    E = Efun(xhat,uu,yy);
    Edx = Efun(xhat+dx*e,uu,yy);
    J(:,j) = (Edx-E)/dx;
end
```

Function to Evaluate Vehicle Dynamics

```
function [x_dot,y] = dynamics(x,u)

global g lf lr h r m Iz c1 c2 c3

nx = length(x);
if nx == 3
    mu = 1;
elseif nx == 4
    mu = x(4);
end

vx = x(1); vy = x(2); wz = x(3);
df = u(1); dr = u(2); wf = u(3); wr = u(4);

vfx = vx*cos(df)+(vy+lf*wz)*sin(df);
vfy = -vx*sin(df)+(vy+lf*wz)*cos(df);
vrx = vx*cos(dr)+(vy-lr*wz)*sin(dr);
vry = -vx*sin(dr)+(vy-lr*wz)*cos(dr);

sfx = (r*wf-vfx)/vfx;
sfy = -vfy/vfx;
srx = (r*wr-vrx)/vrx;
```

```
sry = -vry/vrx;

sf = norm([sfx sfy]);
sr = norm([srx sry]);
if sf == 0
    nf = c1*c2-c3;
else
    nf = (c1*(1-exp(-c2*sf))-c3*sf)/sf;
end
if sr == 0
    nr = c1*c2-c3;
else
    nr = (c1*(1-exp(-c2*sr))-c3*sr)/sr;
end

mfx = mu*nf*sfx;
mfy = mu*nf*sfy;
mrx = mu*nr*srx;
mry = mu*nr*sry;

gxn = lr*(mfx*cos(df)-mfy*sin(df))+lf*(mrx*cos(dr)-mry*sin(dr));
gxd = lf+lr+h*(mfx*cos(df)-mfy*sin(df)-mrx*cos(dr)+mry*sin(dr));
gx = gxn/gxd;
ffz = m*g*(lr-gx*h)/(lf+lr);
frz = m*g*(lf+gx*h)/(lf+lr);

ffx = mfx*ffz;
ffy = mfy*ffz;
frx = mrx*frz;
fry = mry*frz;

fxf = ffx*cos(df)-ffy*sin(df);
fxr = frx*cos(dr)-fry*sin(dr);
fyf = ffx*sin(df)+ffy*cos(df);
fyr = frx*sin(dr)+fry*cos(dr);

xdot1 = (fxf+fxr)/m+vy*wz;
xdot2 = (fyf+fyr)/m-vx*wz;
xdot3 = (lf*fyf-lr*fyr)/Iz;
if nx == 3
    x_dot = [ xdot1 ; xdot2 ; xdot3 ];
elseif nx == 4
    x_dot = [ xdot1 ; xdot2 ; xdot3 ; 0 ];
end

ax = (fxf+fxr)/m;
ay = (fyf+fyr)/m;
y = [ ax ; ay ; wz ];
```

4.3.2 Estimator Based on Zero Slip Assumption

This estimation algorithm is provided as a baseline for comparison. Since it is based on an assumption of zero wheel slip, it is much simpler than the nonlinear least-squares estimation algorithm. For example, this algorithm uses a simple kinematic relationship rather than a dynamic model, and its output can be evaluated

Table 1: Vehicle/Tire/Road Parameter Values

l_f	1.355	m
l_r	1.527	m
h	0.546	m
r	0.329	m
m	1666	kg
I_z	3447	kg m^2
c_1	1.2801	-
c_2	23.99	-
c_3	0.52	-
μ	variable	-

at each sampling instant without iteration. On the other hand, this algorithm requires measurement of yaw rate which is not a required measurement for the nonlinear least-squares algorithm and, more significantly, this algorithm yields inaccurate velocity estimates when wheel slip is large.

```
function [vxhat,vyhat] = ZSA_est(u,wz)

global lf lr r

df = u(1);
dr = u(2);
wf = u(3);
wr = u(4);

vx_fw = r*wf*cos(df);
vy_fw = r*wf*sin(df)-lf*wz;
vx_rw = r*wr*cos(dr);
vy_rw = r*wr*sin(dr)+lr*wz;

vxhat = [vx_fw;vx_rw];
vyhat = [vy_fw;vy_rw];
```

5 Simulation of Velocity Estimation Algorithm

In this section, an initial simulation of the proposed velocity estimation algorithm is provided. The parameter values used to describe the vehicle model and the tire/road model are listed in Table 1. The friction parameters are selected such that $\mu = 1$ corresponds to dry asphalt. All code and graphical plots use SI units.

The transient maneuver simulated here corresponds to an open-loop motion induced by sinusoidal front-wheel steering through a range of $\pm 30°$ with constant wheel speeds and initial relative velocities equal to 3 m/s directed forward. The scale factor describing the road surface switches from $\mu = 1$ to $\mu = 0.5$ after two seconds. The resulting state variable trajectories of the vehicle dynamics model are shown in Fig. 2.

The nonlinear least-squares estimator with $l = 8$, and 20 full-step iterations, is tested in its extended form wherein μ is estimated along with the velocity components to permit automatic adaptation to varying road surfaces. Since μ is supposed to be piecewise-constant in this formulation, changing abruptly only at particular points in time, the extended state equation is taken as $\dot{\mu} = 0$. Since μ is being modeled as the solution of a differential equation, the only way that μ can change values abruptly is to permit impulsive disturbances in the state model. Since such disturbances are not measured, they are analogous to sensor noise and hence lead to nonzero minimum residuals even when all sensors are modeled as ideal sensors. However, the nonzero minimum residuals can be expected to occur only on sampling windows of length l that include the sampling instant at which such a disturbance occurs; otherwise, under an ideal sensor assumption, the

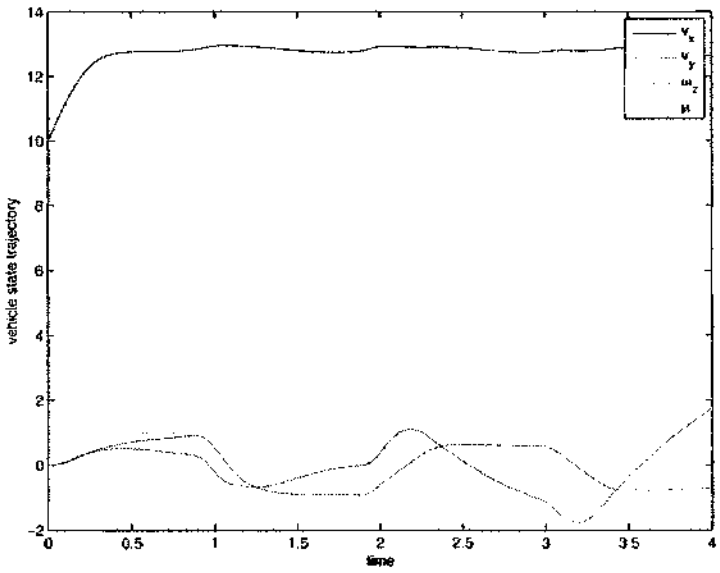


Figure 2: Vehicle response to sinusoidal steering and constant wheel speed inputs, with μ changing from 1 to 0.5 at $t = 2$.

minimum residuals are expected to be zero. The first estimates occur only after the first k-length data set is available. All these expectations are verified by the results shown in Fig. 3.

Finally, for sake of comparison, the method of velocity estimation using zero slip assumptions is also simulated. Each of the two wheels contributes a distinct estimate of the linear velocity vector, and both estimates are shown in Fig. 4. The estimation error is large because the wheel slip is large. The benefit of the nonlinear least-squares estimator is the potential for small estimation error even when wheel slip is large.

A Embedded Load Transfer

Substitution of (28) into (29), with the result substituted into (24) and (25), leads to

$$\begin{bmatrix} f_{Fx} \\ f_{Fy} \end{bmatrix} = \frac{mgl_R - (f_{Fx}\cos\delta_F - f_{Fy}\sin\delta_F + f_{Rx}\cos\delta_R - f_{Ry}\sin\delta_R)h}{l_F + l_R} \begin{bmatrix} \mu_{Fx} \\ \mu_{Fy} \end{bmatrix} \tag{47}$$

Substitution of (28) into (30), with the result substituted into (26) and (27), leads to

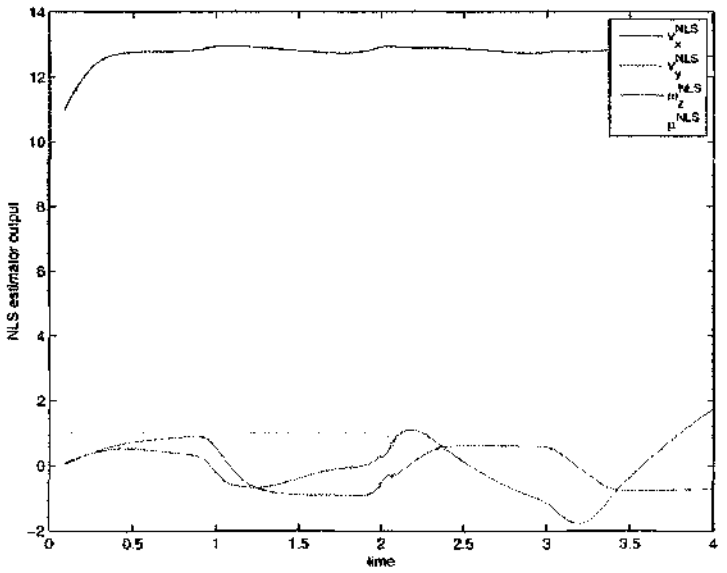
$$\begin{bmatrix} f_{Rx} \\ f_{Ry} \end{bmatrix} = \frac{mgl_F + (f_{Fx}\cos\delta_F - f_{Fy}\sin\delta_F + f_{Rx}\cos\delta_R - f_{Ry}\sin\delta_R)h}{l_F + l_R} \begin{bmatrix} \mu_{Rx} \\ \mu_{Ry} \end{bmatrix} \tag{48}$$

The combination of (47) and (48) may be written in the form

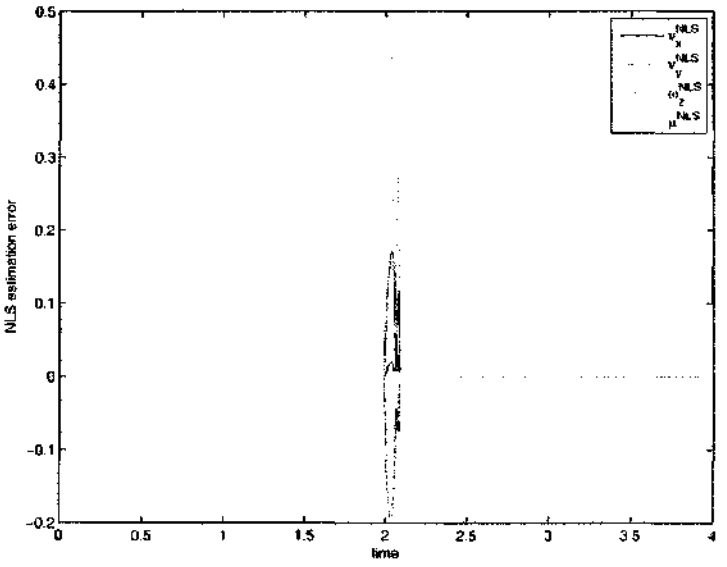
$$\begin{bmatrix} f_{Fx} \\ f_{Fy} \\ f_{Rx} \\ f_{Ry} \end{bmatrix} = \frac{mg}{l_F + l_R} \begin{bmatrix} l_R\mu_{Fx} \\ l_R\mu_{Fy} \\ l_F\mu_{Rx} \\ l_F\mu_{Ry} \end{bmatrix} + \frac{h}{l_F + l_R} \begin{bmatrix} -\mu_{Fx} \\ -\mu_{Fy} \\ \mu_{Rx} \\ \mu_{Ry} \end{bmatrix} \begin{bmatrix} \cos\delta_F & -\sin\delta_F & \cos\delta_R & -\sin\delta_R \end{bmatrix} \begin{bmatrix} f_{Fx} \\ f_{Fy} \\ f_{Rx} \\ f_{Ry} \end{bmatrix} \tag{49}$$

which after further manipulation yields

$$\left(I - \frac{h}{l_F + l_R} \begin{bmatrix} -\mu_{Fx} \\ -\mu_{Fy} \\ \mu_{Rx} \\ \mu_{Ry} \end{bmatrix} \begin{bmatrix} \cos\delta_F & -\sin\delta_F & \cos\delta_R & -\sin\delta_R \end{bmatrix} \right) \begin{bmatrix} f_{Fx} \\ f_{Fy} \\ f_{Rx} \\ f_{Ry} \end{bmatrix} = \frac{mg}{l_F + l_R} \begin{bmatrix} l_R\mu_{Fx} \\ l_R\mu_{Fy} \\ l_F\mu_{Rx} \\ l_F\mu_{Ry} \end{bmatrix} \tag{50}$$

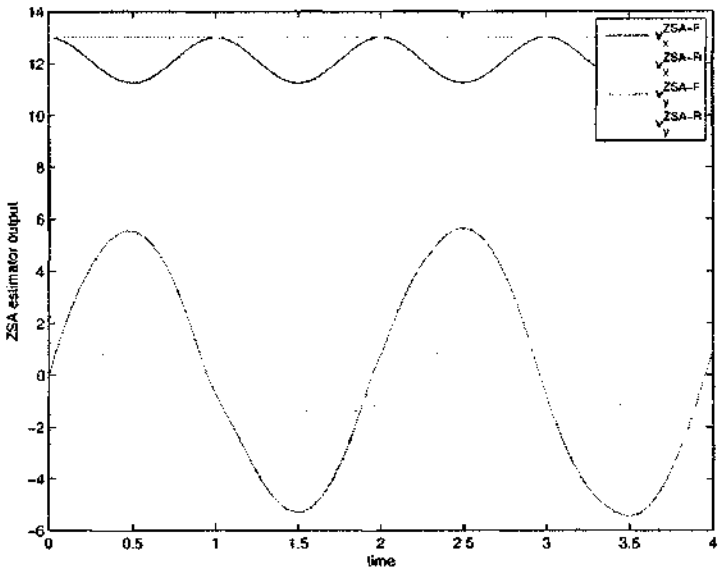


(a) Estimator outputs.

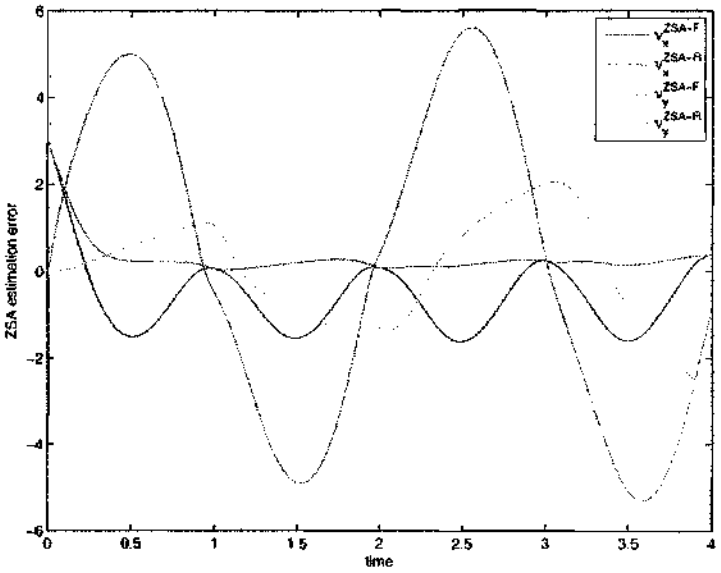


(a) Estimation errors.

Figure 3: Nonlinear least-squares estimator (extended version), assuming measurements of a_x, a_y and ω_z (in addition to wheel speed and steer angle measurements).



(a) Estimator outputs.



(a) Estimation errors.

Figure 4: Zero slip assumption estimator, assuming measurement of ω_z (in addition to wheel speed and steer angle measurements).

Solving this expression for the driving/braking forces yields

$$\begin{bmatrix} f_{Fx} \\ f_{Fy} \\ f_{Rx} \\ f_{Ry} \end{bmatrix} = \frac{mg}{l_F + l_R} \left(I - \frac{h}{l_F + l_R} \begin{bmatrix} -\mu_{Fx} \\ -\mu_{Fy} \\ \mu_{Rx} \\ \mu_{Ry} \end{bmatrix} \begin{bmatrix} \cos\delta_F & -\sin\delta_F & \cos\delta_R & -\sin\delta_R \end{bmatrix} \right)^{-1} \begin{bmatrix} l_R\mu_{Fx} \\ l_R\mu_{Fy} \\ l_F\mu_{Rx} \\ l_F\mu_{Ry} \end{bmatrix} \quad (51)$$

The matrix that must be inverted has a special form, and application of the matrix inversion identity

$$\left(I - xy^T\right)^{-1} = I + \frac{1}{1 - x^T y} xy^T \quad (52)$$

to this specific problem leads to the explicit solution

$$\begin{bmatrix} f_{Fx} \\ f_{Fy} \\ f_{Rx} \\ f_{Ry} \end{bmatrix} = \frac{mg}{l_F + l_R} \left(I + k^{-1} \frac{h}{l_F + l_R} \begin{bmatrix} -\mu_{Fx} \\ -\mu_{Fy} \\ \mu_{Rx} \\ \mu_{Ry} \end{bmatrix} \begin{bmatrix} \cos\delta_F & -\sin\delta_F & \cos\delta_R & -\sin\delta_R \end{bmatrix} \right) \begin{bmatrix} l_R\mu_{Fx} \\ l_R\mu_{Fy} \\ l_F\mu_{Rx} \\ l_F\mu_{Ry} \end{bmatrix} \quad (53)$$

where

$$k = 1 - \frac{h}{l_F + l_R} \begin{bmatrix} -\mu_{Fx} & -\mu_{Fy} & \mu_{Rx} & \mu_{Ry} \end{bmatrix} \begin{bmatrix} \cos\delta_F \\ -\sin\delta_F \\ \cos\delta_R \\ -\sin\delta_R \end{bmatrix} \quad (54)$$

Now that the driving/braking forces are known explicitly in terms of the friction coefficient components, (29) and (30) may be used in conjunction with (28) and (53)–(54) to express the normal forces explicitly in terms of the friction coefficient components as well.

References

[1] M.C. Best, T.J. Gordon and P.J. Dixon, "An extended adaptive Kalman filter for real-time state estimation of vehicle handling dynamics," *Vehicle System Dynamics*, vol. 34, no. 1, pp. 57–75, 2000.

[2] M. Hiemer, A. von Vietinghoff, U. Kiencke and T. Matsunaga, "Determination of the vehicle body side slip angle with non-linear observer strategies," SAE Paper 2005-01-0400.

[3] U. Kiencke and A. Diaß, "Observation of lateral vehicle dynamics," *Control Engineering Practice*, vol. 5, no. 8, pp. 1145–1150, 1997.

[4] L.R. Ray, "Nonlinear state and tire force estimation for advanced vehicle control," *IEEE Transactions on Control Systems Technology*, vol. 3, no. 1, pp. 117–124, 1995.

[5] L.R. Ray, "Nonlinear tire force estimation and road friction identification: simulation and experiments," *Automatica*, vol. 33, no. 10, pp. 1819–1833, 1997.

[6] J. Stephant, A. Charara and D. Meizel, "Virtual sensor: application to vehicle sideslip angle and transversal forces," *IEEE Transactions on Industrial Electronics*, vol. 51, no. 2, pp. 278–289, 2004.

[7] A.Y. Ungoren, H. Peng and H.E. Tseng, "A study on lateral speed estimation methods," *International Journal of Vehicle Autonomous Systems*, vol. 2, no. 1/2, pp. 126–144, 2004.